Lecture-2

- MVC
- Class and Struct
- Enum
- Computed properties
- Closure
- Demo

What is MVC?

- Model-View-Controller
- Design pattern
- Main concept of iOS development
- Used widely in many PL

Model

- What your application does?
- Objects, Logic, Information
- UI independent(Doesn't case about how the app looks)

```
struct CalculatorModel{

func performOperation(_ symbol: String){}

func setOperand(_ operand: Double){}

func getResult(){}

}
```

Controller

- How your application looks?
- Bridge between Model and View

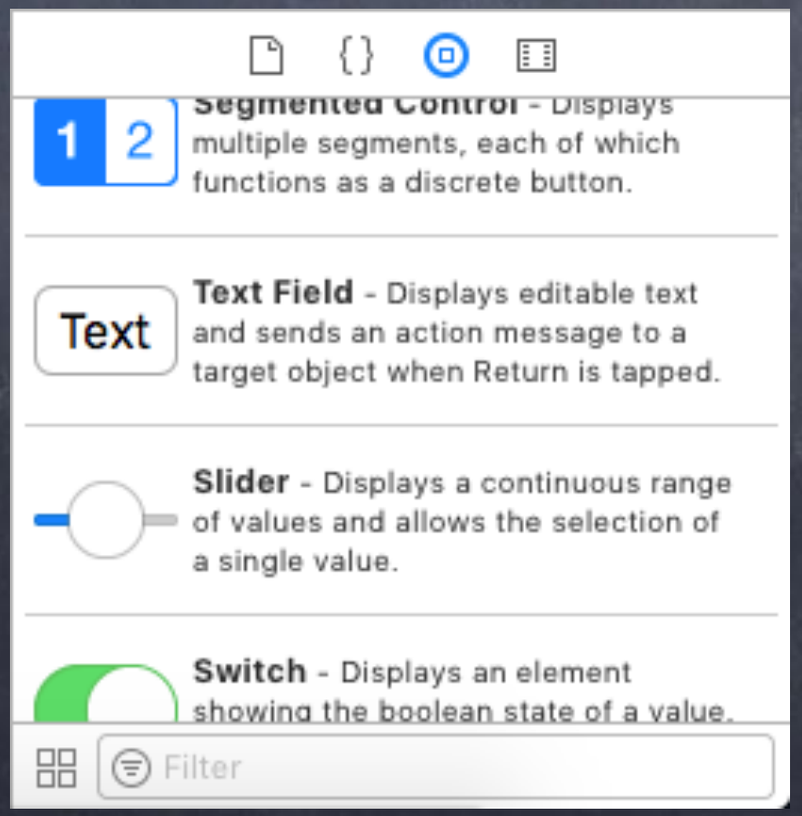
```
class ViewController: UIViewController{

 @IBOutlet weak var myDisplay: UILabel!
 @IBAction func digitPressed(_ sender: UIButton) {}

}
```

View

- Controllers tools(minions), instruments
- UIButton, UITextField, UITableView, UILabel



Model -> View

- Communication is impossible
- Model is UI independent, View is UI dependent

Controller-> Model

- Direct connection from Controller to Model
- Model should have suitable, well-defined API

Model -> Controller

Controller-> View

- Direct connection from Controller to View
- Via Outlets

View-> Controller

- Limited, Blind and Structured communication
- Via Target Action, Delegate, Data Source

Class and Struct

- Class have inheritance, struct don't have
- Class is reference type, Struct is value type
- Other than the above, they are almost the same

Computed property

- Do not actually store a value
- Provide a getter and an optional setter to retrieve and set

```
var displayValue : Double{
 get{}
 set{}
}
```

Enum

- Defines a common type for a group of related values
- Can have associated values of any type
- Optionals are enums

```
var enum Operation{
  case constant(Double)
  case unaryOperation((Double)->Double)
  case binaryOperation((Double,Double)->Double)
  case equals
}
```

Demo